BOOK: Linear Algebra: Theory, Intuition, Code

AUTHOR: Mike X Cohen

WEBSITE: sincxpress.com

CHAPTER: Matrices (chapter 5)

Section 5.1, Interpretations and uses of Matrice

Tratteremo nel libro solo di Matrici 2D

POSSIBILI UTILIZZI

1) trasformazioni lineaeri
2) memorizzazione di derivate parziali
3) sistemi di equazioni
4) memorizzazione di dati

5) regressione per modelli statistici
6) trasformazioni geometriche per computerGrafica
7) memorizzazione di kernels per filtraggio o convoluzione

8) informazioni finanziarie
9) memorizzazione parametri per modelli di diffusione malattie

Section 5.2, Matrix terminology and notation

NOTAZIONI

1) le Matrici sono designate da una lettera maiuscola in grassetto es. $\mathbf{A}$
2) gli Elementi delle matrici sono $a_{i,j}$.

Section 5.3, Matrix dimensionalities

NOTAZIONI

1) Le dimensioni di una Matrice si indicano come: M x N oppure come $\mathbb{R}^{MxN}$, $\mathbb{R}^{MN}$

2) $\mathbb{R}^M$ = serie di vettori colonna
3) $\mathbb{R}^N$ = serie di vettori riga

Section 5.4, code block 5.1 -- The stranspose Operation

TRASPOSIZIONE

si scambiano le righe con le colonne e viceversa

Transposing a vector or matrix

$$\mathbf{B}_{i,j} = \mathbf{A}_{j,i} \qquad (5.1)$$

NOTAZIONI

Matrice trasposta = A^T ___ Trasposta della trasposta = $A^{TT} = A$

In [2]:
```
## import libraries for the entire chapter
import numpy as np
from scipy.linalg import hankel,toeplitz
```

In [3]:
```
# a matrix
A = np.random.randn(2,5)

# two ways to transpose
At1 = A.T
At2 = np.transpose(A)
print(A,' = Matrice iniziale  ---  ')
print(' ')
print(At1,' = Matrice trasposta -- scambio righe-colonne')
```

```
[[ 0.73623868  0.23911593  0.46150235 -1.54852066 -0.21985864]
 [ 2.06171584  0.00740544 -0.4873364  -1.23223828 -0.36861731]]  = Matrice inizia
le  ---

[[ 0.73623868  2.06171584]
 [ 0.23911593  0.00740544]
 [ 0.46150235 -0.4873364 ]
 [-1.54852066 -1.23223828]
 [-0.21985864 -0.36861731]]  = Matrice trasposta -- scambio righe-colonne
```

MATRIX ZOOLOGY:

Section 5.5, code block 5.3 -- Matrix zoology

In [4]:
```
# identity matrix
I = np.eye(4)

# ones matrix
O = np.ones(4)
```

```
# zeros matrix (note the tuple input)
Z = np.zeros((4,4))
```

Section 5.5, Matrix Zoology -- code block 5.5

QUADRATA o RETTANGOLARE

$M = N$ = Matrice quadrata --- $M <> N$ = matrice rettangolare

SIMMETRICA

$\mathbb{A} = \mathbb{A}^T$ = matrice simmetrica

Definitions of a symmetric matrix

$$\mathbf{A} = \mathbf{A}^{\mathrm{T}} \quad (5.3)$$

$$a_{i,j} = a_{j,i} \quad (5.4)$$

SIMMETRICA ### $\mathbb{A} = -\mathbb{A}^T$ = matrice antisimmetrica ### MATRICE IDENTITA' ### $\mathbb{A} * \mathrm{I} = -\mathbb{A}$ dove I = matrice unità ### ZEROS ### 0 = matrice di zeri ### TRANSPOSE ### $\mathbb{A}^T\mathbb{A}$ = " A transpose A " --- una delle Matrici più importanti:

- It is a square matrix, even if $\mathbf{A}$ is rectangular.
- It is symmetric, even if $\mathbf{A}$ isn't.
- It is full-rank if $\mathbf{A}$ is full column-rank.
- It is invertible if $\mathbf{A}$ is full column-rank.
- It has the same row space as $\mathbf{A}$.
- It has orthogonal eigenvectors.
- It is positive (semi)definite.
- It has non-negative, real-valued eigenvalues.
- It is called a "covariance matrix" if $\mathbf{A}$ is a data matrix.
- It often looks pretty (e.g., Figure 5.4).

Section 5.5 - Code-block 5.5

In [5]:
```
# skev matrix (antisimmetrica)
A = np.array([[0,2,3],[-2,0,-5],[-3,5,0]])
# diagonal matrix
D = np.diag([1,2,3,5])

# diagonal elements
R = np.random.randn(3,4)
d = np.diag(R)
print(A,' = matrice antisimmetrica')
print(' ')
print(D,' = matrice diagonale')
# matrice identità
I = np.diag([1,1,1,1])
print(' ')
print(I,' = matrice identità')
O = np.diag([0,0,0,0])
print(' ')
```

```
print(O,' = matrice di zeri')
```

```
[[ 0  2  3]
 [-2  0 -5]
 [-3  5  0]]  = matrice antisimmetrica

[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 5]]  = matrice diagonale

[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]  = matrice identità

[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]  = matrice di zeri
```

Section 5.5, Augmented Matrix -- code block 5.7

MATRICI CONCATENATE

In [6]:
```
# two matrices
A = np.array([[1,4,2],[3,1,9],[4,2,0]])
B = np.array([[7,2],[7,2],[7,1]])

# augmenting
AB = np.concatenate((A,B),axis=1)
print(A,' = Matrice A')
print(' ')
print(B,' = Matrice B')
print(' ')
print(AB,' = Matrice concatenata')
```

```
[[1 4 2]
 [3 1 9]
 [4 2 0]]  = Matrice A

[[7 2]
 [7 2]
 [7 1]]  = Matrice B

[[1 4 2 7 2]
 [3 1 9 7 2]
 [4 2 0 7 1]]  = Matrice concatenata
```

MATRICI TRIANGOLARI

Section 5.5, code block 5.9

In [7]:
```
A =np.random.randint(5,size=(5,5))
# matrix
#A = np.random.rand(5, 5)
# extract the lower triangle
```

```
L = np.tril(A)

# extract the upper triangle
U = np.triu(A)
print(A,' = Matrice di partenza')
print(' ')
print(L,' = Matrice triangolare superiore')
print(' ')
print(U,' = Matrice triangolare inferiore')
```

```
[[0 4 1 1 2]
 [0 2 0 1 0]
 [1 1 1 2 0]
 [1 4 2 3 2]
 [3 2 3 3 3]]  = Matrice di partenza

[[0 0 0 0 0]
 [0 2 0 0 0]
 [1 1 1 0 0]
 [1 4 2 3 0]
 [3 2 3 3 3]]  = Matrice triangolare superiore

[[0 4 1 1 2]
 [0 2 0 1 0]
 [0 0 1 2 0]
 [0 0 0 3 2]
 [0 0 0 0 3]]  = Matrice triangolare inferiore
```

MATRICI DENSE e SPARSE

Una matrice densa ha la maggior parte o tutti gli elementi diversi da zero.
Una matrice sparsa contiene principalmente zeri. È molto efficiente dal punto di vista computazionale.

MATRICI ORTOGONALI

Una matrice è ortogonale se $Q^TQ = I$

In [8]:
```
from scipy.stats import ortho_group
import scipy.linalg
x = ortho_group.rvs(3)     # matrice ortogonale
print(x)
print(' ')
det = np.fabs(scipy.linalg.det(x))
print(det)
print(' ')
xx = np.array([[1,2,2],[2,1,-2],[2,2,-1]])
print(xx)
print(' ')
det2 = np.fabs(scipy.linalg.det(x))
print(det2)
```

```
[[ 0.25393349  0.33227562 -0.90835604]
 [-0.45969535 -0.78482972 -0.41559896]
 [ 0.85099822 -0.52310154  0.04654899]]

1.0000000000000002

[[ 1  2  2]
 [ 2  1 -2]
 [ 2  2 -1]]

1.0000000000000002
```

Section 5.5, code block 5.11

MATRICI TOEPLITZ

In una Matrice Toepltz i valori nelle singole diagonali sono uguali.
È dunque possiblie crearla partendo da un opportuno Vettore

MATRICI HANKEL

In una Matrice Hankel i valori nelle singole anti-diagonali sono uguali.
È dunque possiblie crearla partendo da un opportuno Vettore

In [9]:
```
# the vector
t = [1,2,3,4]

# the matrices (functions imported from scipy at top of script)
T = toeplitz(t)
print(T,' = matrice Toeplitz')
print(' ')
H = hankel(t,r=[2,3,4,1])
print(H,' = matrice Hankel')
```

```
[[1 2 3 4]
 [2 1 2 3]
 [3 2 1 2]
 [4 3 2 1]]  = matrice Toeplitz

[[1 2 3 4]
 [2 3 4 3]
 [3 4 3 4]
 [4 3 4 1]]  = matrice Hankel
```

Section 5.6, code block 5.11 -- Matrix addition and subtraction

ADDIZIONE e SOTTRAZIONE di MATRICI

In [10]:
```
a = np.array([[1,2],[3,4]])
b = np.array([[1,1],[1,1]])
c = a+b
print(c)
```

```
[[2 3]
 [4 5]]
```

Section 5.7, code block 5.11 -- Scalar-matrix multiplication

PRODOTTO di uno SCALARE per una MATRICE

In [11]:
```
a = np.array([[1,2],[3,4]])
lambda1 = 2
c = lambda1*a
print(c)
```

```
[[2 4]
 [6 8]]
```

Section 5.8, code block 5.13 -- "Shifting" a matrix

"SPOSTAMENTO" di una MATRICE

Ad una matrice aggiungo un arbitrario λ*I,

In [12]:
```
# lambda
l = .01

# I
I = np.eye(4)

# the shifted matrix
A = np.random.randn(4,4)
As = A + l*I
print(A)
print(' ')
print(As)
```

```
[[-0.81481652  0.90200776 -0.93081745 -0.97191008]
 [ 1.89965853 -0.22364689 -1.0376687   0.09467423]
 [-0.60649398 -2.48928408  0.17835065  0.72481127]
 [ 0.45121033  0.21467706 -0.88312874  0.67392786]]

[[-0.80481652  0.90200776 -0.93081745 -0.97191008]
 [ 1.89965853 -0.21364689 -1.0376687   0.09467423]
 [-0.60649398 -2.48928408  0.18835065  0.72481127]
 [ 0.45121033  0.21467706 -0.88312874  0.68392786]]
```

Section 5.9, code block 5.15 -- Diagonal and trace

estrazione DIAGONALE e "TRACCIA"

La diagonale è un vettore che contiene gli elementi della diagonale

"Traccia" è un numero sintetico che estraggo dalla matrice --- si applica solo a Matrici quadrate

Trace is the sum of diagonal elements

$$tr(A) = \sum_{i=1}^{M} a_{i,i} \qquad (5.15)$$

In [13]:
```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
v = np.diag(A)
tr = np.trace(A)
print(v)
print(tr)
```

```
[1 5 9]
15
```

Section 5.10, Exercises

Section 5.11, Answers

Section 5.12, Code challenges

Section 5.13, code block 5.17 -- Code solutions

In [14]:
```
# two matrices
A = np.random.randn(4,2)
B = np.random.randn(4,2)

# initialize the output
C = np.zeros((2,2))

# loop over columns and compute dot products
for coli in range(2):
    for colj in range(2):
        C[coli,colj] = np.dot(A[:,coli],B[:,colj])
```

Section 5.13, code block 5.19

ESERCIZI - rendere simmetrica una matrice

In [15]:
```
# a full matrix
A = np.random.randn(4,4)

# get its lower triangle
Al = np.tril(A)

# add it to its transpose
S = Al + Al.T
```

Section 5.13, code block 5.21

ESERCIZI - indicizzare la diagonale

In [16]:
```
# initialize
D = np.zeros((4,8))

# loop over smaller dimension
for d in range(min(D.shape)):
    D[d,d] = d+1
D
```

Out[16]:
```
array([[1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 2., 0., 0., 0., 0., 0., 0.],
       [0., 0., 3., 0., 0., 0., 0., 0.],
       [0., 0., 0., 4., 0., 0., 0., 0.]])
```